Survey of Key-Recovery Attacks on LowMC in a Single Plaintext/Ciphertext Scenario

Lorenzo Grassi[1,2], Daniel Kales[2], Christian Rechberger[2], and Markus Schofnegger[2]

[1]University of Nijmegen, Nijmegen, The Netherlands,
L.Grassi@cs.ru.nl
[2]Graz University of Technology, Graz, Austria,
{firstname.lastname}@iaik.tugraz.at

May 13, 2020

One of the main use cases of the cipher LowMC is in public-key signature schemes like Picnic as it helps to keep the signature size small. The relevant setting here is that only a single (plaintext, ciphertext) pair is available to the attacker.

Even though LowMC received considerable cryptanalytic attention, this very restrictive setting rules out many attack vectors. This paper surveys the remaining known analysis technique that can still be applied and their impact on round-reduced LowMC. We hope that this is a useful starting point for more in-depth cryptanalysis and also serves as a base-line for the LowMC-Cryptanalysis Challenge.

1 Introduction

LowMC [Alb+15; Alb+16] is a flexible and very parameterizable block cipher construction, aiming at various use cases that benefit from its low number of multiplications.

Some of these use cases have requirements that are very different from standard ciphers like AES. In the case of the public-key signature scheme Picnic [Cha+17], key-recovery or forgery attacks can be reduced to attacks on the underlying one-way function for which

LowMC is proposed to be used. While LowMC attracted various efforts with cryptanalytic relevance since its formal publication in 2015, e.g. [Alb+16; DEM15; Din+19; Din+15; RST18], attacks in such a setting are generally rare. One notable example is an attack on 4-round AES with one known plaintext: [BDF11]. In the rest of this note, we will give survey techniques that can be used to analyze LowMC in such a setting as well.

2 Brief description of LowMC

LowMC is a block cipher using a substitution-permutation network. It allows the user to freely choose a selection of parameters, such as the block size n and the key size κ, the number of S-boxes, and the allowed data complexity d of attacks, where $d = \log_2(n_{\text{pairs}})$ and n_{pairs} refers to the number of (plaintext, ciphertext) pairs needed for an attack. The S-boxes of LowMC have a size of 3 bits and the number of S-boxes can be reduced in order to (potentially) reduce the number of multiplications. LowMC also uses a partial S-box layer, which is not common for block ciphers with substitution-permutation networks (Zorro [Gér+13] is an exception), but which is a design approach that has recently been adopted by various other constructions, for example in [Gra+19; Gra+].

One round of LowMC is shown in Fig. 1. The output of each S-box $S(\cdot)$ is defined as

$$S(a, b, c) = (a + (b \cdot c), a + b + (a \cdot c), a + b + c + (a \cdot b)),$$

where a, b and c are bits, and additions and multiplications are executed in $\mathbb{F}_2$. The m S-boxes are used for the first $3m$ bits of the n-bit state, while the remaining $n - 3m$ bits (for $n > 3m$) are passed to the affine layer unchanged.

Key addition and constant addition take place after the affine layer in $\mathbb{F}_2$, where the round keys and the n-bit round constants are chosen during the instantiation of the cipher and then fixed. More precisely, each round key is generated by multiplying the initial key with a randomly chosen $n \times \kappa$ binary matrix of rank $\min(n, \kappa)$.

The affine layer itself consists of a multiplication in $\mathbb{F}_2$ of the current state with an $n \times n$ binary matrix, which is different for each round and chosen randomly out of all invertible $n \times n$ matrices. All of these matrices are randomly chosen and fixed during the instantiation of LowMC, thus generation takes place in constant time. The pseudorandom bits needed for the matrices are generated by the Grain LFSR [HJM07].

Instances of LowMC Considered

With a view towards the use in the Picnic signature scheme, we consider the following variants of LowMC, where N denotes the block size (and key size) and s denotes the number of S-boxes:

- LowMC with $N = 128$ and $s \in \{1, 10\}$

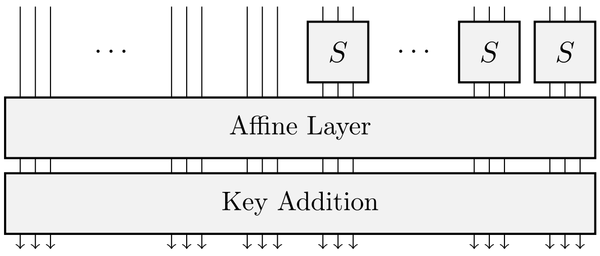

Figure 1: One round of LowMC using a partial nonlinear layer.

- LowMC with $N = 192$ and $s \in \{1, 10\}$
- LowMC with $N = 256$ and $s \in \{1, 10\}$
- LowMC with $N = 129$ and $s = 43$
- LowMC with $N = 192$ and $s = 64$
- LowMC with $N = 255$ and $s = 85$

Note that the nonlinear layer is partial in the first three variants, which is also shown in Fig. 1. In the final three variants, the nonlinear layer is full, which makes these versions much more similar to traditional ciphers like AES. We illustrate in Fig. 2 how these constructions are built.

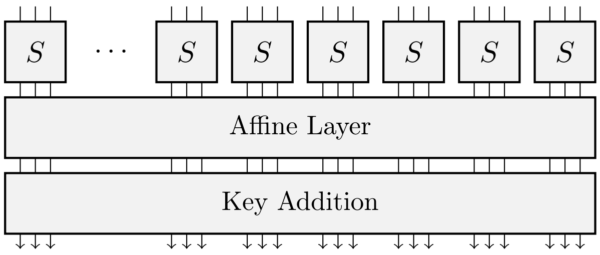

Figure 2: One round of LowMC using a full nonlinear layer.

3 Survey of Selected Techniques

We continue to survey a selection of techniques that can be used to recover a key of LowMC in the restrictive low-data scenario of Picnic, where only a single (plaintext, ciphertext) pair is available to the attacker.

3.1 Attacks Using Gröbner Bases

In an attack using Gröbner bases, the attacker first describes the construction by a system of equations. Subsequently, a Gröbner basis [Buc65; CLO97] for the ideal defined by the corresponding polynomials is calculated and finally used to solve for specified variables. In more detail, the attacks consist of three steps:

1. Set up an equation system and compute a Gröbner basis using approaches such as Buchberger's algorithm [Buc65], F4 [Fau99], or F5 [Fau02].
2. Perform a change of term ordering for the computed basis using an algorithm like FGLM [Fau+93].
3. Solve the univariate equation for the last variable using a polynomial factoring algorithm, and substitute into other equations to obtain the full solution of the system.

From an attacker's point of view it is necessary to evaluate all three of these steps, and to conclude that an attack works if and only if the sum of all three complexities is below the complexity of a brute force attack. However, as a designer, it is sufficient to make one of these complexities prohibitively high. Designers usually use the first step of the attack to make security claims about their constructions, since this is in many cases the most expensive step. Hence, we will focus on the complexity for setting up a Gröbner basis.

Cost of Computing a Gröbner Basis. Given a generic system of n_e polynomial equations

$$F_1(x_1, \dots, x_{n_v}) = F_2(x_1, \dots, x_{n_v}) = \cdots = F_{n_e}(x_1, \dots, x_{n_v}) = 0$$

in n_v variables $x_1, \dots, x_{n_v}$, we expect the computation of a Gröbner basis [BFP12] to be

$$\mathcal{C}_{\mathrm{GB}} \in \mathcal{O}\left(\binom{n_v + D_{\mathrm{reg}}}{D_{\mathrm{reg}}}^{\omega}\right), \tag{1}$$

where $2 \leq \omega < 3$ is the linear algebra exponent representing the complexity of matrix multiplication and D_{reg} is the degree of regularity. The constants hidden by $\mathcal{O}(\cdot)$ are relatively small, which is why $\binom{n_v+D_{\mathrm{reg}}}{D_{\mathrm{reg}}}^{\omega}$ is typically used directly. In general, computing the degree of regularity is a hard problem. However, the degree of regularity for "regular sequences" [Bar+05] is given by

$$D_{\mathrm{reg}} = 1 + \sum_{i=1}^{n_e} (d_i - 1), \tag{2}$$

where d_i is the degree of F_i. In regular sequences, the number of equations is the same as the number of variables. More generally, for *semi-regular sequences* (the generalization of regular sequences to $n_e > n_v$) the degree of regularity can be computed as the index of the first non-positive coefficient in

$$H(z) = \frac{1}{(1-z)^{n_v}} \times \prod_{i=1}^{n_e} (1 - z^{d_i}).$$

It is believed that random systems behave like semi-regular systems with high probability. Hence, assuming the equations in our system have no additional structure, the complexity of computing a Gröbner basis depends on the number of equations, the degrees of these equations, and on the number of variables.

We use Eq. (1) and Eq. (2) to estimate the cost of our attacks. Every strategy is first shortly described, and finally a table containing the complexities is given.

3.1.1 Strategy 1

In this strategy, we describe each key bit as a boolean variable, i.e., $k_i \in \{0,1\}, 0 \leq i < n$. This amounts to n variables for an n-bit instance of LowMC, using the fact that every round key is a linear combination of the master key. Additionally, we introduce variables at the output of each nonlinear layer. LowMC can be instantiated with a variable number s of 3-bit S-boxes in each round, hence we need to add $3 \cdot s \cdot (r-1)$ variables, where r denotes the number of rounds. Note that we can omit the variables for the last S-box layer, since the output of these S-boxes is described by the known output bits.

Regarding the equations, we map the $3 \cdot s$ output variables of each S-box layer to the n input variables of the previous layer (right before the application of the S-boxes). Since each S-box in LowMC has an algebraic degree of 2, this results in $3 \cdot s$ quadratic boolean equations for $(r-1)$ rounds. In the last round, we use the known ciphertext bits and relate them to the output of the previous layer, for which we need n quadratic boolean equations.

In total, we have the following number of variables n_v and number of equations n_e:

$$\begin{aligned} n_v &= 3 \cdot s \cdot (r-1) + n, \\ n_e &= 3 \cdot s \cdot (r-1) + n. \end{aligned}$$

Since $n_e = \alpha n_v, \alpha = 1$, we can estimate the complexity of the attack using e.g. the results given in [Bar+13]. Using the most optimistic bound, we obtain a complexity of $\mathcal{O}(2^{(1-0.208)\cdot n_v})$, which suggests that

$$r \geq \frac{n - 0.792n}{2.376s} + 1$$

rounds are sufficient to provide security against this attack.

MS: The effect of the linear layer is completely ignored here!

3.1.2 Strategy 2

Instead of using the results given in [Bar+13], we can also take the standard approach to calculate the complexities. The number of variables and the original number of equations is the same as in the first strategy, but now we add more equations in order to specifiy that we are working with boolean variables (note that this has also been done implicitely in the first strategy). In more detail, we add

$$\begin{aligned} f_{n_e+1} &= x_1 - {x_1}^2, \\ f_{n_e+2} &= x_2 - {x_2}^2, \\ &\vdots \\ f_{n_e+n_v} &= x_{n_v} - {x_{n_v}}^2 \end{aligned}$$

to our original system, obtaining $2n_v = n_v$.

3.1.3 Strategy 3

For the LowMC instance using $n = 192$ and $s = 1$ or $s = 10$, note that $\frac{192}{3}, \frac{3}{3}, \frac{30}{3} \in \mathbb{N}$. We can therefore use polynomials over $\mathbb{F}_{2^3}$ instead of working over $(\mathbb{F}_2)^3$ for each S-box. This results in

$$\begin{aligned} n_v &= s \cdot (r-1) + \frac{n}{3}, \\ n_e &= s \cdot (r-1) + \frac{n}{3}, \end{aligned}$$

where again $n_v = n_e$. We assume a valid (linear) translation of the original linear layer, and we also assume that the S-box with algebraic degree 2 can be converted into an S-box with a word-level degree of 3, i.e., our n_e equations are of degree 3.

Alternatively, note that also $\frac{192}{6}, \frac{30}{6} \in \mathbb{N}$, which means we can also work over $\mathbb{F}_{2^6}$.

3.1.4 Guessing Key Bits – First Option

Another possibility is to split the attack into a brute-force part and a Gröbner basis computation. Specifically, we can e.g. use the first strategy and guess $\hat{k}$ bits of the key. For a particular guess, the Gröbner basis attack proceeds like described, but with only $n_v - \hat{k}$ variables. The complexity is then in $\mathcal{O}(2^{\hat{k}} \cdot \Gamma)$, where Γ is the complexity of the Gröbner basis attack on $n_v - \hat{k}$ variables. Note that this is now an overdetermined system, and again it is hard to estimate the resulting complexity in the boolean case. For the second and the third strategy, it is much easier.

This approach does not exploit the fact that we can build a linear system of equations by guessing key bits, a strategy which will be described next.

3.1.5 Guessing Key Bits – Second Option

The most competitive version of the attacks presented here is probably a combination of a linear equation system and a degree-d equation system ($d > 1$) which will be solved using a Gröbner basis.

For example, consider the 128-bit version of LowMC using $s = 1$ S-box in each round. If we guess the three input key bits to the S-box in each round, we have a 5-dimensional system of $3 \cdot 41 = 123$ linear equations in 128 key variables $k_0, k_1, \dots, k_{127}$ after 41 rounds. We can therefore solve this system for 123 variables (solving linear equation systems is fast) and use a Gröbner basis computation for the last round, using the remaining 5 variables and 128 equations of degree 2 for the ciphertext bits.

In general, the number of variables and equations for the linear system when describing r_1 rounds is

$$\begin{aligned} n_{v,l} &= n, \\ n_{e,l} &= 3 \cdot r_1, \end{aligned}$$

and the number of variables and equations for the quadratic system when describing the last r_2 rounds and guessing $\hat{k}$ key bits is

$$\begin{aligned} n_{v,q} &= (n - \hat{k}) + (r_2 - 1) \cdot 3s, \\ n_{e,q} &= n + (r_2 - 1) \cdot 3s. \end{aligned}$$

Note that we only use a Gröbner basis computation for the last r_2 rounds. It remains to find a good tradeoff between r_1 and r_2 in order to maximize $r = r_1 + r_2$ for a complexity $\in \mathcal{O}(2^{n-\varepsilon})$.

3.1.6 Higher Degrees vs. Fewer Variables

In many situations, increasing the degrees of the equations and decreasing the number of variables can lead to a reduced complexity. For the strategies proposed above, this would mean that we do not introduce variables at every new round, but instead, for example, at every second round. In general, if we skip $\hat{r}$ rounds, the degree-d equations reach a degree of $d \cdot \hat{r}$. It is easy to estimate resulting complexities for the second and third strategy, but not for the first one. For strategies 2 and 3, we can observe an increase of the complexity when we skip any rounds, so we set $\hat{r} = 0$ in our calculations.

Attack	n	s	r	C
Strategy 1	128	1	13	130
		10	3	149
	192	1	18	192
		10	3	200
	256	1	24	257
		10	4	274
Strategy 2	128	1	1	145
		10	1	145
	192	1	1	204
		10	1	204
	256	1	1	263
		10	1	263
Strategy 3, $\mathbb{F}_{2^3}$	192	1	1	346
		10	1	346
Strategy 3, $\mathbb{F}_{2^6}$	192	10	2	198

Table 1: Complexities (in bits) of attacks using the various strategies, without skipping any variables or guessing any key bits.

3.1.7 Complexities

The attack complexities for the strategies described above are given in Table 1, where we assume that variables are introduced in each round and that no key bits are guessed (i.e., $\hat{r} = \hat{k} = 0$).

Note that for every combination where $r = 1$ rounds are enough, no variables are introduced. This means that, apart from the equations, the attack complexity depends only on the number of variables needed for the key. We can therefore try to guess part of the key, and the results for the best values of $\hat{k}$ are given in Table 2. Note that values for the first strategy are missing, because it is unclear whether the results still apply for $\alpha \gg 1$ (in $n_e = \alpha n_v$).

However, we can also use the second option of key guessing, which allows to skip many initial rounds by guessing parts of the subkey. The resulting complexities are given in Table 3, where r_1 and r_2 are chosen such that $r = r_1 + r_2$ is maximized and C is just below n. We can observe that r_2 is always close to 1, mainly because the Gröbner basis computation for one round is already more expensive than guessing the intermediate key bits.

Potentially, an attacker can use the linear equations for the first r_1 rounds to add an additional r_1 rounds to the attack. In this case, the number of rounds $r \approx r_1$ should be doubled. Using $r = (r_1 + r_2) \cdot 2 + 1$, this results in the recommendation given in Table 4.

Attack	n	s	$\hat{k}$	r	C
Strategy 2	128	1	127	1	130
		10	127	1	130
	192	1	191	1	194
		10	191	1	194
	256	1	255	1	258
		10	255	1	258
Strategy 3, $\mathbb{F}_{2^3}$	192	1	102	8	193
		10	102	2	199
Strategy 3, $\mathbb{F}_{2^6}$	192	10	42	6	209

Table 2: Complexities (in bits) of attacks using the various strategies, where $\hat{k}$ key bits are guessed using the first option.

Attack	n	s	$\hat{k}$	r_1	r_2	r	C
Strategy 2	128	1	114	38	1	39	127.81
		10	90	3	1	4	123.54
	192	1	177	59	1	60	191.17
		10	150	5	1	6	184.63
	256	1	240	80	2	82	255.43
		10	240	8	1	9	254.51
Strategy 3, $\mathbb{F}_{2^3}$	192	1	183	61	1	62	191.64
		10	180	6	1	7	190.26
Strategy 3, $\mathbb{F}_{2^6}$	192	10	180	6	1	7	186.64

Table 3: Complexities (in bits) of attacks using the various strategies, where $\hat{k}$ key bits are guessed using the second option.

Note though that this considers only the "standard" way of computing a Gröbner basis in the remaining variables, and maybe a more efficient approach can be taken using SAT solving techniques (or even using results from [Bar+13] or [JV17]). A more conservative approach would therefore be to choose $r = (r_1 + r_2) \cdot 3 + 1$.

3.2 Approaches Using MitM Attacks

Besides providing a starting point for estimating the security of LowMC against Gröbner basis attacks, we also used a MitM tool to experimentally evaluate LowMC with full nonlinear layers, where $N = 192$ and $s = 64$. Specifically, we used a tool proposed in 2016 [DF16] which tries to simplify a given equation system to subsequently mount a

n	s	$r = (r_1 + r_2) \cdot 2 + 1$
128	1	79
	10	9
192	1	121
	10	13
256	1	165
	10	19

Table 4: Recommendation for round numbers considering the Gröbner basis approaches.

MitM attack on a given cipher. The first step is thus to represent the cipher as a system of equations over a field. For our purposes, we work over $\mathbb{F}_2$.

To illustrate how these equation systems look like, we give here an example for LowMC with $N = 6$ and $s = 2$, where one round is used:

$$\begin{gathered}
P[0] + K_0[0] = X_0[0], P[1] + K_0[1] = X_0[1], \\
P[2] + K_0[2] = X_0[2], P[3] + K_0[3] = X_0[3], \\
P[4] + K_0[4] = X_0[4], P[5] + K_0[5] = X_0[5], \\
Y_0[0] = S_0(X_0[0], X_0[1], X_0[2]), Y_0[1] = S_1(X_0[0], X_0[1], X_0[2]), \\
Y_0[2] = S_2(X_0[0], X_0[1], X_0[2]), Y_0[3] = S_0(X_0[3], X_0[4], X_0[5]), \\
Y_0[4] = S_1(X_0[3], X_0[4], X_0[5]), Y_0[5] = S_2(X_0[3], X_0[4], X_0[5]), \\
Y_0[0] + Y_0[1] + Y_0[3] + Y_0[4] = Z_0[0] + 1, Y_0[1] + Y_0[4] + Y_0[5] = Z_0[1] + 1, \\
Y_0[1] + Y_0[3] + Y_0[4] + Y_0[5] = Z_0[2], Y_0[1] + Y_0[2] + Y_0[5] = Z_0[3] + 1, \\
Y_0[2] + Y_0[3] + Y_0[5] = Z_0[4], Y_0[0] + Y_0[2] + Y_0[3] + Y_0[5] = Z_0[5] + 1, \\
K_0[2] + K_0[4] + K_0[5] = K_1[0], K_0[0] + K_0[2] + K_0[3] + K_0[4] = K_1[1], \\
K_0[0] + K_0[1] = K_1[2], K_0[4] = K_1[3], \\
K_0[0] + K_0[2] + K_0[4] = K_1[4], K_0[1] + K_0[2] + K_0[3] + K_0[4] + K_0[5] = K_1[5], \\
C[0] + K_1[0] = Z_0[0], C[1] + K_1[1] = Z_0[1], \\
C[2] + K_1[2] = Z_0[2], C[3] + K_1[3] = Z_0[3], \\
C[4] + K_1[4] = Z_0[4], C[5] + K_1[5] = Z_0[5].
\end{gathered}$$

In this equation sytem, $P[\cdot]$ denotes a plaintext bit, $K[\cdot]$ denotes a key bit, and $C[\cdot]$ denotes a ciphertext bit. Plaintext bits and ciphertext bits are known, key bits are unknown. X_i denotes the state after key addition in round i, Y_i denotes the state after the nonlinear layer in round i, and Z_i denotes the state after the linear layer in round i. The subkey in round i is denoted by K_i. For simplicity, the matrices for this system were generated using the random engine built into `Sage`, whereas we used the proposed matrices generated by the Grain LFSR for our tests against the full version.

The tool quickly reported attacks on a 2-round version. We could not find any results on a 3-round version, and it reported that it could not find attacks on 4 rounds of LowMC. We thus believe that 4 rounds of LowMC provide security against this attack vector in the case of a full S-box layer.

3.3 Approaches Using Coding Theory and the Number of Nonlinear Gates

In [Zaj17], an attack approach using coding theory is presented, with results applicable to the solving complexity of generic constructions using a certain number of unknowns. In particular, an upper bound is given for the complexity of solving random systems. In the case of LowMC, this bound states that the number of `AND` gates has to be at least $3.27 \cdot |k|$, where $|k|$ denotes the bit length of the secret key k. Hence,

$$r \cdot 3s \geq 3.27n \implies r \geq \frac{3.27n}{3s}.$$

Using this result, lower bounds for the numbers of rounds needed in our cases are given in Table 5. Note that when using a full S-box layer, the number of rounds is always 4 if the key size is equal to the block size.

n	s	$r = \left\lceil \frac{3.27n}{3s} \right\rceil$
128	1	140
	10	14
192	1	210
	10	21
256	1	280
	10	28

Table 5: Recommendation for round numbers using the approach given in [Zaj17].

3.3.1 Experimental Results

To further analyze the runtime of this approach, we implemented it for small-scale variants of LowMC. We give a short summary of the attack approach by [Zaj17] and report on our experimental results.

Modeling LowMC as an MRHS Equation System. The first step is to model the LowMC encryption circuit as a multiple right-hand side (MRHS) equation system. A MRHS system is similar to a traditional linear equation system defined by $A \cdot x = b$, however, the right-hand side is a set of possible solutions S so that $A \cdot x \in S$. We build a small MRHS equation system for each AND gate in the circuit as follows. Declare an unknown variable for all key variables and also for each AND gate declare an unknown variable for its output. Then model each AND gate in on the encryption circuit as a MRHS equation system with the solution set $S = \{(0,0,0),(0,1,0),(1,0,0),(1,1,1)\}$, i.e., all valid assignments of an AND gate. Any affine constants are added to the corresponding columns in S. The plaintext is also treated as an affine constant, whereas the ciphertext is used to cancel up to n unknown variables via linear algebra. This results in a MRHS equation system with a matrix $M \in \mathbb{F}_2^{\mu \times 3\mu}$ and a solution set S with $|S| = 4^{\mu}$, where μ is the number of AND gates in the encryption circuit.

Transforming the MRHS Equation System Into a Decoding Problem The next step is interpreting the matrix M as the generator matrix of a binary linear code and calculating its parity check matrix $H \in \mathbb{F}_2^{2\mu \times 3\mu}$. Each original solution x of the MRHS system produces a codeword $c = xM$, such that $c \in S$, and furthermore due to the properties of the parity check matrix we have $cH^T = 0$. Zajac [Zaj17] describes two approaches to ensure we find a c that is in S, and therefore corresponds to a valid solution x of the original MRHS equation system. We follow his second approach, which results in an augmented matrix $Q \in \mathbb{F}_2^{3\mu \times 2\mu}$ such that $vQ = q$, a classical syndrome decoding problem.

Solving the Syndrome Decoding Problem. We follow the approach of Zajac and use a modified version of the Lee-Brickel decoding algorithm to find the solution v. It randomly permutes the 3-bit blocks of Q to get Q' and then randomly assigns columns of Q' to distinct parts H_1, H_2, H_3 of a bigger matrix $H = (H_1|H_2|H_3)$. It then follows the standard Lee-Brickel algorithm to bring $(H|q)$ into a form of $(I|U|\hat{q})$ and tests all combinations of 2 columns of U, checking if their hamming weight is less than or equal to the number of allowed errors minus 2. If so, the rest of the errors can be cancelled by picking columns from I. However, an additional step needs to be added, since we have to ensure that no vectors u_i are from the same original block in Q, otherwise this would not be a valid solution of the original MRHS equation system. This additional rejection rate increases the runtime to the original syndrome decoding algorithm.

Experimental Results. We implemented this attack for small-scale variants of LowMC, and attacked a 15-bit and 21-bit, and additionally compared this to brute-forcing the key. The round numbers are chosen so that the instances have a security level of n bits. We have based this implementation on the original implementation[1] of Zajac, which is written in `sagemath`, and for fairness, we have also implemented the brute-force of LowMC in `sagemath`. Both attacks were run on a standard desktop computer equipped with an Intel i7-4970 CPU and 16 GB of RAM, on a single thread each. We report the results in Table 6, where the runtimes of the decoding attack are the average of 100 runs and the brute-force time is exactly half of the time to brute-force the full key-space. We can see that except for the trivially insecure 2 round variants, the time for the decoding attack always exceeds the brute-force time, in contrast to Zajac's original experiments

[1] `https://github.com/zajacpa/DecodingAttack`

on a reduced variant of SIMON. However, we note that this reduced variant of SIMON was not chosen in a way that it provides k bits of security, where k is the length of the secret key, whereas these instances of LowMC are chosen in this way.

Our implementation of the decoding attack on LowMC is available on GitHub[2]. This repository contains all necessary files and instructions on how to use the program.

n	s	r	Decoding Attack (s) mean (median)	Brute-force Attack (s)
15	5	2	1.39 (0.84)	4.85
15	5	3	270.35 (82.10)	6.81
15	**5**	**4**	19258 (10248)	8.41
21	7	2	294.64 (50.20)	437.83
21	7	3	17992 (15513)	582.46
21	**7**	**4**	- (-)	741.54

Table 6: Experimental results on attacking small-scale variants of LowMC with a decoding attack. Bold instances are chosen to provide n bits of security. "-" indicates the implementation did not finish in one week of runtime for our trials.

4 Conclusions and Open Problems

Other Approaches Using Gröbner Bases. In this survey, we describe various strategies on how equations systems for the permutation may be built and finally simplified and solved using Gröbner basis techniques. We emphasize, however, that these equation systems can be represented in many different ways. For example, instead of considering single rounds, an attacker may want to consider two or three consecutive rounds. Using this approach, they may be able to reduce the number of unknowns and equations, while simultanously increasing the degrees and the complexity of the equations.

Application of Cryptanalysis for Rasta. Rasta[Dob+18] borrows some of the design elements of LowMC and together with other ideas significantly improves implementation characteristics in use-cases involving homomorphic encryption. While it does not improve implementation properties when plugged into a signature scheme like Picnic, it's analysis setting is closely related and hence analysis techniques developed for it [Dob+18; Dob+20] will be relevant for LowMC in the single plaintext/ciphertext setting as well.

[2] `https://github.com/lowmcchallenge/lowmcchallenge-material`

Improvements to and Variants of the Decoding Attack. The decoding attack in Section 3.3 has the best theoretical runtimes, however, we did not manage to produce a performant attack in practice. An interesting open problem would be to integrate modern decoding algorithms into this approach and also perform a theoretical analysis of the overhead of this attack compared to the plain decoding algorithm. Additionally, one could build a MRHS equation system not based on the AND-gates of the cipher, but rather on the SBox-level. This leads to a linear code with differnt parameters, however, in our preliminary experiments, the runtime of this approach was even slower.

References

[Alb+15] Martin R. Albrecht, Christian Rechberger, Thomas Schneider, et al. "Ciphers for MPC and FHE". In: *EUROCRYPT (1)*. Vol. 9056. Lecture Notes in Computer Science. Springer, 2015, pp. 430–454 (cit. on p. 1).

[Alb+16] Martin Albrecht, Christian Rechberger, Thomas Schneider, et al. *Ciphers for MPC and FHE*. Cryptology ePrint Archive, Report 2016/687. `https://eprint.iacr.org/2016/687`. 2016 (cit. on pp. 1, 2).

[Bar+05] M Bardet, JC Faugere, B Salvy, et al. "Asymptotic behaviour of the index of regularity of quadratic semi-regular polynomial systems". In: *The Effective Methods in Algebraic Geometry Conference (MEGA)*. 2005, pp. 1–14 (cit. on p. 4).

[Bar+13] Magali Bardet, Jean-Charles Faugère, Bruno Salvy, et al. "On the complexity of solving quadratic Boolean systems". In: *J. Complexity* 29.1 (2013), pp. 53–75 (cit. on pp. 5, 6, 9).

[BDF11] Charles Bouillaguet, Patrick Derbez, and Pierre-Alain Fouque. "Automatic Search of Attacks on Round-Reduced AES and Applications". In: *CRYPTO*. Vol. 6841. Lecture Notes in Computer Science. Springer, 2011, pp. 169–187 (cit. on p. 2).

[BFP12] Luk Bettale, Jean-Charles Faugère, and Ludovic Perret. "Solving polynomial systems over finite fields: improved analysis of the hybrid approach". In: *International Symposium on Symbolic and Algebraic Computation, ISSAC'12*. ACM, 2012, pp. 67–74 (cit. on p. 4).

[Buc65] Bruno Buchberger. "Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal". PhD thesis. University of Innsbruck, 1965 (cit. on pp. 3, 4).

[Cha+17] Melissa Chase, David Derler, Steven Goldfeder, et al. "Post-Quantum Zero-Knowledge and Signatures from Symmetric-Key Primitives". In: *ACM Conference on Computer and Communications Security*. ACM, 2017, pp. 1825–1842 (cit. on p. 1).

[CLO97] David A. Cox, John Little, and Donal O'Shea. *Ideals, Varieties, and Algorithms – An Introduction to Computational Algebraic Geometry and Commutative Algebra.* 2nd ed. Undergraduate Texts in Mathematics. Springer, 1997 (cit. on p. 3).

[DEM15] Christoph Dobraunig, Maria Eichlseder, and Florian Mendel. "Higher-Order Cryptanalysis of LowMC". In: *ICISC 2015.* Vol. 9558. 2015, pp. 87–101 (cit. on p. 2).

[DF16] Patrick Derbez and Pierre-Alain Fouque. "Automatic Search of Meet-in-the-Middle and Impossible Differential Attacks". In: *CRYPTO (2).* Vol. 9815. Lecture Notes in Computer Science. Springer, 2016, pp. 157–184 (cit. on p. 9).

[Din+15] Itai Dinur, Yunwen Liu, Willi Meier, et al. "Optimized Interpolation Attacks on LowMC". In: *ASIACRYPT 2015.* Vol. 9453. LNCS. 2015, pp. 535–560 (cit. on p. 2).

[Din+19] Itai Dinur, Daniel Kales, Angela Promitzer, et al. "Linear Equivalence of Block Ciphers with Partial Non-Linear Layers: Application to LowMC". In: *EUROCRYPT 2019.* Vol. 11476. LNCS. 2019, pp. 343–372 (cit. on p. 2).

[Dob+18] Christoph Dobraunig, Maria Eichlseder, Lorenzo Grassi, et al. "Rasta: A Cipher with Low ANDdepth and Few ANDs per Bit". In: *CRYPTO 2018.* Vol. 10991. 2018, pp. 662–692 (cit. on p. 13).

[Dob+20] Christoph Dobraunig, Farokhlagha Moazami, Christian Rechberger, et al. "Framework for faster key search using related-key higher-order differential properties: applications to Agrasta". In: *IET Information Security* 14.2 (2020), pp. 202–209. DOI: `10.1049/iet-ifs.2019.0326` (cit. on p. 13).

[Fau+93] Jean-Charles Faugère, Patrizia M. Gianni, Daniel Lazard, et al. "Efficient Computation of Zero-Dimensional Gröbner Bases by Change of Ordering". In: *J. Symb. Comput.* 16.4 (1993), pp. 329–344 (cit. on p. 4).

[Fau02] Jean-Charles Faugère. "A new efficient algorithm for computing Gröbner bases without reduction to zero (F5)". In: *Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation ISSAC.* Ed. by T. Mora. isbn: 1-58113-484-3. ACM Press, July 2002, pp. 75–83 (cit. on p. 4).

[Fau99] Jean-Charles Faugere. "A new efficient algorithm for computing Gröbner bases (F4)". In: *Journal of Pure and Applied Algebra* 139.1-3 (1999), pp. 61–88 (cit. on p. 4).

[Gra+] Lorenzo Grassi, Reinhard Lüftenegger, Christian Rechberger, et al. "On a Generalization of Substitution-Permutation Networks: The HADES Design Strategy". In: *Eurocrypt 2020.* Ed. by Anne Canteaut and Yuval Ishai, pp. 674–704 (cit. on p. 2).

[Gra+19] Lorenzo Grassi, Daniel Kales, Dmitry Khovratovich, et al. "Starkad and Poseidon: New Hash Functions for Zero Knowledge Proof Systems". In: *IACR Cryptology ePrint Archive* 2019 (2019), p. 458 (cit. on p. 2).

[Gér+13] Benoît Gérard, Vincent Grosso, María Naya-Plasencia, et al. "Block Ciphers That Are Easier to Mask: How Far Can We Go?" In: *CHES 2013*. Vol. 8086. LNCS. 2013, pp. 383–399 (cit. on p. 2).

[HJM07] Martin Hell, Thomas Johansson, and Willi Meier. "Grain: a stream cipher for constrained environments". In: *IJWMC* 2.1 (2007), pp. 86–93 (cit. on p. 2).

[JV17] Antoine Joux and Vanessa Vitse. "A Crossbred Algorithm for Solving Boolean Polynomial Systems". In: *NuTMiC*. Vol. 10737. Lecture Notes in Computer Science. Springer, 2017, pp. 3–21 (cit. on p. 9).

[RST18] Christian Rechberger, Hadi Soleimany, and Tyge Tiessen. "Cryptanalysis of Low-Data Instances of Full LowMCv2". In: *IACR Trans. Symmetric Cryptol.* 2018.3 (2018), pp. 163–181 (cit. on p. 2).

[Zaj17] Pavol Zajac. "Upper bounds on the complexity of algebraic cryptanalysis of ciphers with a low multiplicative complexity". In: *Des. Codes Cryptogr.* 82.1-2 (2017), pp. 43–56 (cit. on pp. 11, 12).